A STANDARD METHOD FOR SURFACE FITTING BY ORTHOGONAL POLYNOMIALS

by

E. Kantorowitz

Technical Report No. 12

March 1971
A STANDARD METHOD FOR SURFACE FITTING BY ORTHOGONAL POLYNOMIALS

by E. Kantorowitz

Abstract

An algorithm for least squares fitting of a polynomial to a function \( z(x,y) \) of two independent variables is described. It is assumed that the function is given by a number of "measured values", which may be arbitrarily distributed over the \( xy \)-plane. The polynomial is constructed as a linear combination of a set of polynomials which are orthonormalized over the set of given data points by the modified Gram Schmidt procedure. Formulas for the estimation of time and space required for the calculations and for the accuracy obtained are given. It is demonstrated that the method is numerically robust, that the costs of computations are reasonable, and that the procedure is suitable as a standard tool for least squares surface fitting. The experiments employed for the practical evaluation of the procedure are discussed.

Introduction

In many applications, functions given by "measured values" are often encountered. In order to be able to manipulate such a function, it is approximated by a simple mathematical expression. This paper is concerned with approximation by polynomials of functions of two independent variables. A function

\[ z(x,y) \]

is thus given by \( \{(x_T, y_T, z_T, w_T) | r=1,2,...,N_p\} \), where \( z_T \) is the value measured at \( (x_T, y_T) \) and \( w_T \) is the weight assigned to this measurement. It is assumed that the measured values are not accurate, and the coefficients of the approximating polynomial \( P(x,y) \) are therefore calculated by employing the criterion of least squares. Since \( z(x,y) \) may be considered as a representation of a surface, the term "surface fitting" is commonly used for this problem. If the \( (x_T, y_T) \)'s happen to be the points of intersection of a rectangular grid in the xy-plane, the algorithm of Clark Kubik and Phillips [2] may be used. This method is economic in terms of computer time and storage as compared to the general procedure discussed in this paper. Another economic method by Clenshaw and Hayes [3] may be used in the somewhat more general case, where the \( (x_T, y_T) \) are positioned on a number of parallel
lines, such that the number of points on each of these lines is "adequate".
The number and relative position of the data points may with this method
vary from line to line. In the general case, where the \((x_r, y_r)\) are arbitrarily
distributed over the xy-plane, the procedure apr 2 \([7]\) is recommended.
This procedure has been intensively employed since 1962 for fitting of poly­
nomials to parts of ship surfaces \([6]\) and for some other applications. This
paper presents the results of an experimental study of the accuracy obtained
with the method.

The apr2 Algorithm
The given function \(z(x,y)\) is fitted by the polynomial
\[
P(x,y) = \sum_{i=0}^{X_e} \sum_{j=0}^{Y_e} K_{ij} x^i y^j
\]

This polynomial is generated as a linear combination of a set of
\[
q = (X_e+1)(Y_e+1)
\]

orthonormal polynomials \(Q_t(x,y)\), i.e.
\[
P(x,y) = \sum_{t=1}^{Q} C_t Q_t(x,y)
\]

The method is similar to that of Cadwell and Williams \([1]\), though a different
set of polynomials is employed here. Furthermore, apr2 employs the modified
Gram Schmidt procedure as described in Rice's paper \([10]\) for the generation
of the \(Q_t\) polynomials, and it may therefore be expected to yield more accurate
results than those obtained by the method of \([1]\), where the (unmodified)
Gram Schmidt procedure is employed.
The \( Q_t \) polynomials are generated in a sequence which is illustrated by Fig. 1. Any \( Q_t \) is composed of all the terms whose \( x \) and \( y \) exponent correspond to the previously generated \( Q_t \). \( Q_5 \) in Fig. 1, for instance, is composed of terms with the \((x,y)\) exponents \((0,0)\) \((1,0)\) \((0,1)\) \((2,0)\) and \((1,1)\). Any \( Q_t \) is characterized by its degree \( m \), and the \( y \)-exponent \( n \) of its last term. For \( Q_5 \), for instance, \((m,n)=(2,1)\). The first approximation to a \( Q_t \) is chosen as

\[
Q_t^{(1)} = x^{m-n} y^n
\]  

(4)

The \( N_p \) values of \( Q_t^{(1)} \) at the given data points constitute a vector \( u_t^{(1)} \). i.e.

\[
u_t^{(1)} = x^{m-n} y^n , \quad r = 1, \ldots, N_p
\]  

(5)

The previously generated orthonormal polynomials \( \{Q_i | i=1,\ldots,t-1 \} \) are also represented by vectors \( \{v_i | i=1,\ldots,t-1 \} \). \( v_t \) is now calculated by the process:

\[
u_t^{(j+1)} = u_t^{(j)} - (u_t^{(j)}, v_j) v_j \quad \text{for } j=1,\ldots,t-1
\]  

(6)

\[v_t = u_t^{(t)} / ||u_t^{(t)}||
\]
This algorithm is numerically equivalent to the modified Gram Schmidt procedure as described by Rice [10], but it has the advantage that the number of vectors is not fixed at the beginning, and it is possible at any stage to add another vector to the set.

The coefficients \( C_t \) in (3) are calculated as

\[
C_t = (v_t, z - \sum_{i=1}^{t-1} C_i v_i)
\]

..... (7)

where the vector \( z \) is composed of the \( N_p \) "measured" \( z_t \)-values. By employing (7) instead of \( C_t = (v_t, z) \), a considerable gain in accuracy is obtained.

In the ALGOL 60 procedure apr 2 [7], the coefficients of the polynomials \( Q_t \) are stored in array \( L \). The vectors \( v_t \) are stored in the array \( z_q \), while the vector

\[
\sum_{i=1}^{t-1} C_i v_i
\]

is stored in array \( z_k \).

Computer Costs

The computation time for apr 2 equals

\[
C N_p q^2 = C f q^3
\]

..... (8)

\( C \) being a constant and \( f = N_p / q \). This formula has been verified experimentally and the numerical values of \( C \) as measured on 3 different computers are stated in a comment in the procedure apr 2 [7]. The storage requirement is

\[
q N_p + q^2 / 2 + \text{< smaller terms >} = (f + 0.5) q^2
\]

..... (9)

These storage and time requirements are not extremely low, but they resulted in reasonable costs in the practical cases encountered.
Accuracy

Experiments were made in order to estimate the magnitude of the inaccuracies in the polynomials calculated by apr 2, which stem from the round off errors in the computations. The experiments were made with "nice" functions, which are believed to be quite representative for most applications. The "nice" functions employed were the "test polynomials".

\[ P(x,y) = \sum_{i=0}^{X_e} \sum_{j=0}^{Y_e} \frac{(i+1)(j+1)}{2q} x^i y^j \]  

(11)

where \( q = (x_e+1)(y_e+1) \). These polynomials are characterized by "modest" variations within the test square \(-1 \leq x,y \leq 1\). Their integral

\[ \int_{-1}^{1} \int_{-1}^{1} P(x,y) \, dx \, dy = \left[ \frac{x_e+1}{2} \right] \left[ \frac{y_e+1}{2} \right] \frac{2}{q} = \frac{1}{2} \]  

(12)

Polynomials were chosen in order to avoid the problem of whether the function may at all be approximated by a polynomial, which is a different question.

In the experiments, an adequate number \( N_p \) of points \((x_T,y_T)\) were picked within the test square. Furthermore, weights \( w_T \) were chosen at random in the interval \([0;2]\) and assigned to the points. The values \( z_T = P(x_T,y_T) \) were then calculated by \((11)\), and a new polynomial of the same degree was fitted by apr 2 to the set of \((x_T,y_T,z_T,w_T)\) values. The differences between the numerical values of the corresponding coefficients of the original and calculated polynomial represent the error generated through the process. Actually, relative errors \( d_t \) were calculated, i.e., the difference between the true and calculated coefficient were divided by the true coefficient. A "standard deviation \( E \) was then calculated as

\[ E = \sqrt{\frac{1}{q} \sum_{t=1}^{q} \frac{d_t^2}{(q-1)}} \]  

(13)
The data points were chosen by one of two schemes:

1. Randomly picked within the test square. The number $N_p$ of data points to be picked was chosen after an experiment, where the same polynomial was calculated employing different values, for $N_p$ (Table 1).

<table>
<thead>
<tr>
<th>Distribution</th>
<th>1.0</th>
<th>1.2</th>
<th>1.4</th>
<th>1.6</th>
<th>1.8</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
<th>6.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>1$\times10^{-3}$</td>
<td>3$\times10^{-3}$</td>
<td>5$\times10^{-4}$</td>
<td>1$\times10^{-3}$</td>
<td>2$\times10^{-4}$</td>
<td>4$\times10^{-4}$</td>
<td>5$\times10^{-4}$</td>
<td>1$\times10^{-4}$</td>
<td>1$\times10^{-4}$</td>
</tr>
<tr>
<td>Mesh points of</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rectangular grid</td>
<td>4$\times10^{-4}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Standard error in the calculated coefficients as a function of $f = N_p/q$. The experiment was made for $(X_e,Y_e) = (6,6)$ on an IBM 360/50 employing a 24 bit mantissa (roughly 7 decimal digits).

In the case shown in Table 1, poor or relatively poor accuracy was obtained for $f = 1.0, 1.2$ and 1.6. With a large number of data points ($f > 2$), the chances for "bad luck" with the set of randomly picked points is small, and a "maximal" error, which in our case is of the magnitude of error of $10^{-4}$, was observed. The same "maximal" accuracy was obtained for this case with $f = 1$, when the data points were chosen as:

2. the mesh points of the rectangular grid shown in Fig. 2. This distribution of points seems to be "optimal" in the sense of obtaining the "maximal" accuracy with the least number of points. This scheme is economical on the computer because of the low number of points, and was therefore used for the rest of the experiments.

Rice [10] shows that the modified Gram Schmidt procedure is quite advantageous, but that even with this method the degree of nonorthogonality increases with the number of polynomials $q$. It may therefore be expected that the accuracy obtained by apr 2 is a decreasing function of $q$. This was confirmed by the experiments, which are summarized in Fig. 3.
Fig. 2 - Grid of data points. The points marked 0 are employed in the case of the polynomial \((X_e,Y_e) = (3,4)\), and \(f=1\). The rule is that the number of grid lines parallel to the x and y axis are \([(Y_e+1)(f^2 + \frac{1}{2})]\) and \([(X_e+1)(f^2 + \frac{1}{2})]\) respectively. The points marked x are denoted as "points between the data points" and are employed along with the data points for the calculation of the standard deviation between the values of the true and the calculated test polynomial. The standard deviations calculated for these two sets of points proved to be of nearly the same magnitude.

The coefficients of the test polynomial were thus calculated with about

\[
a - 0.6 - \frac{1}{20} q
\]

correct decimal digits, where \(a\) is the number of decimal digits which correspond to the number of bits \(N\) in the mantissa of the floating point numbers employed \((a = (N-1)\log_{10}2)\). Formula (14) did not fit in the two cases \(q=64, 81\) for 24 bits mantissa (Fig. 3), where the errors in the calculated coefficients were quite high, while the errors in the values of the corresponding polynomial were still in accordance with (15). A repeated calculation of these two cases with twice as many data points \((f=2)\), produced more accurate coefficients in close agreement with (14). The values of the test polynomial were calculated with about

\[
a - 0.6 - \frac{1}{50} q
\]

correct decimal digits (Fig. 3). The observed maximal errors in the calculated polynomials were 2-5 times the values given by (14) and (15).

The Terms included in the polynomial
The polynomial terms generated by apr 2 are illustrated by Fig. 1, and those generated by the procedure described by Cadwell and Williams [1] in Fig. 4.
Fig. 3 - Standard deviation $E$ in the calculated polynomials. $E$ was calculated by (13) and represent a relative error.

* IBM 360/50, 24 bits mantissa (6.9 dec. digits), $f=1$.  
* IBM 360/50, 24 bits mantissa (6.9 dec. digits), $f=2$.  
* ELLIOTT 503, 30 bits mantissa (8.7 dec. digits), $f=1$.  
* IBM 360/50, 56 bits mantissa (16.6 dec. digits), $f=1$.  

The lines in the diagrams represent formulas (14) and (15).

<table>
<thead>
<tr>
<th>$X_{\text{exp}}$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_{\text{exp}}$</td>
<td>t=1</td>
<td>t=3</td>
<td>6</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>t=2</td>
<td>5</td>
<td>9</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>8</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4 - Sequence of generation of orthogonal polynomials in the Cadwell and Williams procedure. $t$ denotes the sequence number.
It is obviously possible to devise additional schemes which will generate other sets of polynomial terms. The Algol 60 procedure apr 2 of [7] may be modified to any of these schemes by two minor changes. The first is to give $X_e$ and $Y_e$ such values that there will be room in array $K$ for all the polynomial coefficients, and the second is to modify the \texttt{for} m \texttt{for} n \texttt{...} clause in accordance with the scheme to be employed. The computer time and storage required for any of these schemes may be evaluated from (8) and (9), where $q$ denotes the actual number of polynomial coefficients. For the Cadwell and William scheme $q$ thus becomes

$$q = \frac{(X_e+2)(Y_e+1)}{2} \quad \text{...... (16)}$$

where $X_e = Y_e$ is the degree of the polynomial. apr 2 has in some experiments been modified to produce a polynomial employing Cadwell and Williams scheme, and the accuracies observed were in agreement with (14) and (15) when $q$ is calculated by (16). This version of apr 2 is not identical with Cadwell and Williams original procedure [1] as it employs the modified Gram Schmidt method. This experiment indicates, however, that (14) and (15) are valid for any scheme of polynomial generation employed in apr 2.

It is often difficult to decide which scheme for generation of polynomial terms and which degree of polynomial are the most suitable for a given "measured function". The Cadwell and Williams scheme (Fig. 4) has the advantage that there is only one parameter, the degree of the polynomial, to be varied when searching for the optimal solution. The computer may be programmed to increase the degree of the polynomial $X_e$ by 1 as long as

$$g \cdot \sum_{t=1}^{h} (c_{t}v_{t} - z)^2 > h \cdot \sum_{t=1}^{g} (c_{t}v_{t} - z)^2 \quad \text{...... (17)}$$

where $g$ and $h$ are the $t$-values corresponding to the polynomial of the degree $X_e$ and $X_e+1$. The criterion (17) does not necessarily produce the optimal solution. A more sophisticated algorithm is employed by Makinson in [8]. (17) may also be used for other schemes of polynomial generation. $g$ may, for instance, be given the values $1, 2, 3, \ldots$ and $h$ may be chosen as $g+1$. 

Technion - Computer Science Department - Technical Report CS0012 - 1971
In apr 2 procedure of [7] the maximum exponents of x and y, \( X_e \) and \( Y_e \), may be different and this scheme is therefore advantageous when the "degree of variation" in the x and y directions are different. apr 2 is especially useful in an iterative process where the computer at each stage plots a picture of the surface, which is then used by the scientist to make a guess of "better" \( X_e \) and \( Y_e \) values. This technique is possible when some of the curves of intersection between the surface and a number of planes x=constant and y=constant are somehow produced in the plotted picture. These curves are to be approximated by polynomials of degrees \( \leq Y_e \) and \( \leq X_e \) respectively. The task is thus to guess \( X_e \) and \( Y_e \) on the basis of a visual study of the shape of these plotted curves. With some experience this may be quite easily done. Such a process combines the power of the computer with the intuition and knowledge of the scientist and has proved to be efficient [6] and [11]. The graphical representation has furthermore proved useful for detection of errors in the measured values, and for giving the scientist a feel of the character of the function.

Conclusions
apr 2 seems to yield quite accurate results when employed on "nice" functions. The accuracies observed are given by Fig. 3, (14), and (15). The computer time and storage required may be evaluated by (8) and (9), and seem to be reasonable.

For randomly distributed data points full accuracy was observed when the number of points \( N \) was larger than twice the number of coefficients in the approximating polynomials (Table 1). For data points ordered in a rectangular grid (Fig. 2) full accuracy could in most cases be obtained with \( f=1 \).

A close guess of the degree of the polynomial which provides the best approximation to a given function may often be made on the basis of a visual study of a graphical representation of the surface.

The errors in the values of the polynomials calculated by apr 2 at the data points and between them were of the same magnitude of order. These errors could be calculated by (15) even in two extreme cases, where the errors in the coefficients of the calculated polynomial were considerably greater than evaluated by (14).
Repeated calculations of the same polynomial, but with different sets of data points weights produced close results. Furthermore, by employing randomly picked weights full reproducability of the experiments may only be obtained by specifying the weights employed. It is therefore recommended to use the weight 1 for all points in similar experiments.

Formulas (14) and (15) give the accuracy of the calculated polynomial as a function of the length of the floating point mantissa employed, and is therefore applicable for any computer. These formulas take however no account of differences in the implementation of the floating point arithmetic in different machines. It may thus be seen from Fig. 3 that the Elliott 503 performs slightly better than may be expected from the formulas, which fit closer to the IBM 360/50. One of the possible reasons is that the normalization shift in the IBM 360/50 is only made in steps of 4 bits, a limitation which does not exist in the Elliott 503. Experiments of the type described in this paper may thus also be employed for comparing the performance of the floating point arithmetic in different computers.

The Algol 60 procedure apr 2 of [7] may easily be modified to produce polynomials having terms with other x and y exponents than those obtained with the scheme illustrated by Fig. 1. The computer time and storage required in any of these cases may be evaluated by (8) and (9), when \( g \) denotes the number of terms in the actual polynomial. There is an indication that (14) and (15) are also applicable in these cases for estimation of the accuracy of the results.
REFERENCES


